Homework One and Two: hw2: Discuss in section today
- Introduction to C
- Review of basic programming principles
- Building from “fgetc” and “fputc”
- Input and output strings and numbers
- Introduction to pointers
- Emphasize limits (arrays bounds)

Hello World:

//Java:
public class Driver {
    public static void main (String [] args) {


    }
}

C++:
#include <iostream>
using namespace std;
int main (int argc, char * const * argv) {

}

C:
#include <stdio.h>
int main (int argc, char * const * argv) {

}
Standard output: The regular display output by your program.
Java:
C++:
C:

Display is ________: ____ displayed immediately to the user. Certain events must happen for the display to show:
1.
2.
3.
4.
5.

When is display to standard output lost:
Ex: When is the following output lost:
   fprintf (stdout, Hello World”);
Ans:
1.
2.
3.
**Standard error:** The display for ____________________________
Java:
C++:
C:

Display is _______: _____ displayed immediately to the user
When is display to standard output lost:
Ex: When is the following output lost:
    fprintf (stderr, “Hello World”);
Ans:
    1.

**Standard input:** The normal input from ______ for ______ applications.
Java:
C++:
C:

Buffered input: (2 buffers: 1 – keyboard buffer, 2 – standard input buffer)
Keyboard buffer: processes all input (backspaces…finish keyboard input: ‘\n’)

Certain event must happen for the input to get into the standard input buffer:
   1. ______________________________________

Ex: ‘A’ - ‘A’ goes in the ______ buffer, standard input is ____________
‘B’ - ‘B’ goes in the _______ buffer, standard input is ____________
‘C’ - ‘C’ goes in the _______ buffer, standard input is ____________
<ret> - ______ goes into the ______ buffer, keyboard buffer is ________.

How to get input in C:
   char character = NULL;
   
   while (character != ‘\n’) {     // <ret>
       character = fgetc (stdin);
   }


Standard input is _______ when program begins…therefore, a call to fgets when stdin is empty caused your program to: __________.

Once standard input has contents, then “fgets” will
______________________________ and it will return
______________________________.

Exception: Control D input
1) It’s the input to end all test cases in the autograder.
2) Your execution score will be 0 if you fail to handle ^D.
3) It’s the indication that you are done with your application.
4) To enter “End of File” also known as “EOF.”
5) Side note: ^D is in UNIX what ^Z is on Windows. (Do not ^Z on UNIX).
Works only when stdin is __________
Input is:
  ^D
Not:
  123^D
  123^D<ret>
  ^D<ret>
### Numbering Systems

<table>
<thead>
<tr>
<th></th>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Octal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>Base</td>
<td>Base</td>
<td>Base</td>
<td>Not representable</td>
</tr>
<tr>
<td>prefix needed</td>
<td>prefix required</td>
<td>prefix required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example:

Hexadecimal: Mapping of binary...easy way to ____________________

32 bits = 1 word (size of int, long (C/C++)...reading direct contents of memory.

Largest Number in 32 bits:
- Most significant bit (left most bit: 0 = positive, 1 = negative)

Largest Number in 32 bits:
- unsigned number: all bits are _______________...no ____ bit

The Apple Analogy:
1 distinct plate of 10 decimal apples
another distinct plate of 012 octal apples
another distinct plate of 1010 binary apples
another distinct plate of 0xa hexadecimal apples
Put them all in one bag...how many do you have?
Answer: _____ apples!

ASCII Codes: American Standard Codes for Information Interchange
- ______________ displayed to the user
- ______________ input from the keyboard
Example keyboard input:
“123\n” …
Example string in C/C++
“123\n” …

NULL = ‘\0’ = 0 = 0x0 = the last character in every ____________________.
Newline = ‘\n’ = 10 = 0xa = the last character ____________________. 
Numeric input:

C: unsigned long ul;  // always positive, 32 bits of magnitude
    signed long sl;  // 1 sign bit, 31 bits of magnitude

The computer represents negative numbers in a notation called:
    Two’s complement
Two’s complement is both an algorithm and a representation
    - How:
        o Binary number: flip all bits
        o Add one
Ex:
    1 in decimal is 0000000000….000001 in 32 bits in binary
How do you represent -1 in two’s complement?

Here’s how:
    00000000….00000001: start with a positive number

    11111111….11111110: flip
        +1: add one

-----------------------------
Ans:

Ex:
    Given an unknown negative number, what is its magnitude?

    11111111….11111111: -???

    00000000….00000000: flip
        +1: add one

-----------------------------
Therefore: going positive to negative, negative to positive is the

\[
\begin{align*}
11111111...11111111: & \quad -1 \\
+ 00000000...0000001: & \quad +1 \\
\end{align*}
\]

Observation: ________________________________________.
Is this a problem?

Ans:_____________________.

Ex in 4 bits, signed addition: 1 bit of sign, 3 bits of magnitude

\[
\begin{align*}
0101: \quad & \quad 5 \\
+ 0110: \quad & \quad +6 \\
\end{align*}
\]

Problem: Overflow…a bit of magnitude __________ or _____.

\[
\begin{align*}
0100: \quad & \quad \text{flip} \\
+ \ 1: \quad & \quad \text{add one} \\
\end{align*}
\]

Observation: overflow occurred:

\[
\begin{align*}
\circ & \quad \_______________ \\
\circ & \quad \_______________ \\
\circ & \quad \_______________ \\
\end{align*}
\]

When a computer program produces overflow…what happens?

Ans:

Ramifications: ____________________________.
Layout of memory: (You need to know where are the variables that you use, how long they exist, what other variables are nearby).

**Text:** Where your program resides while it is executing.
- Function/methods definitions, while loops, if statements, constant strings
- Read only

**Data:**
- global memory, static variables
- created when your program is loaded into memory
- go away when your program ends
- 0 initialized

**Stack:** (Run-Time Stack – RTS)
- local variables, parameters, return results (user controlled)
- old registers (whatever that is) (by the operating system)
- return addresses (whatever that is) (by the operating system)
- created when your function/method is called
- go away when your function/method ends
- uninitialized – initialized with contents from the last use of space

**Heap:**
- dynamic memory (memory allocated at run-time)
- C: ”malloc” (m:memory, alloc: allocate), “free” frees up the memory
- C++: “new” creates, “delete” frees up the memory
- Java: “new” creates, garbage collector frees up the memory
- C/C++ if you don’t free your memory… **memory leak…bad thing!**